Goals and Objectives

This module describes our objectives for the whole project.

Motivation

The following equation provides the basic model for a blurry observation:

$$f = Ku + \omega$$

The FTVd algorithm quickly reconstructs a clear image from a noisy, blurry image for a given blur kernel. Our objective is to study the efficacy of the Fast Total Variation deconvolution (FTVd) algorithm for recovering images affected by common types of blur.

Problem

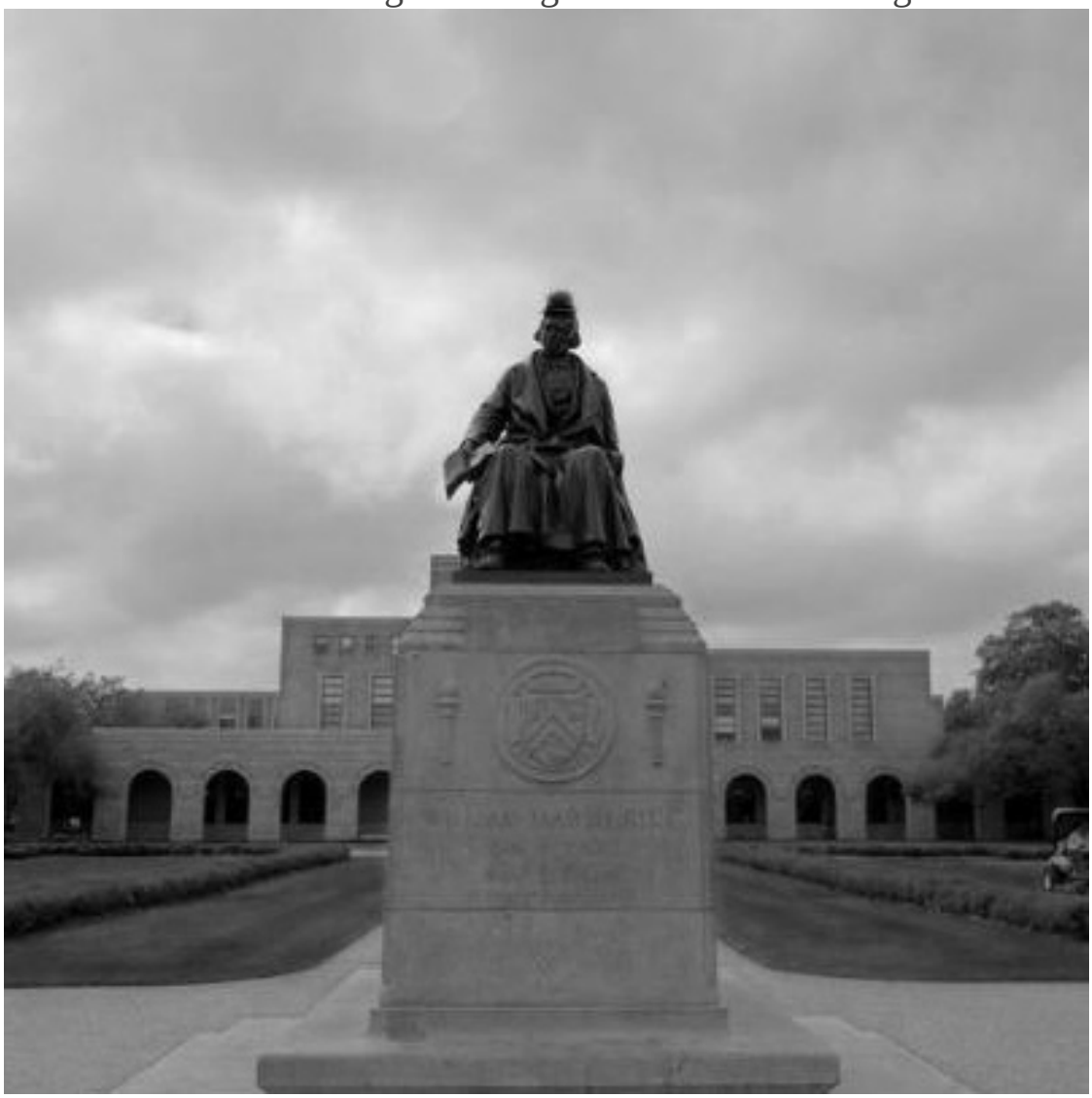
While blur can be used as an artistic effect, much of photography distills down to practices that minimize blur. For instance, there are techniques for holding the camera steady to minimize shake or movement of the focal plane. High shutter speeds are used to “freeze” object motion. Furthermore, manufacturers design high-end lenses with “image stabilization” or “vibration reduction” features, and some design the sensor in the camera body to resist forms of camera shake. But as cameras become increasingly integrated with small form factor devices, their size and lack of heft make them more susceptible to blur caused by camera shake or movement between the shooter and the subject. As a result, the techniques and equipment that are used in digital SLR photography present a less practical solution to a growing photography community.

Even with the right equipment, however, blur is an unavoidable reality in images. Also, that same integrated circuits and small devices in that same equipment will introduce different kinds of noise that further corrupt the received signal. For a photographer trying to capture the moment in which an event occurs, a blurry image can be an opportunity lost forever. For the photographer shooting a dim scene without a steady hand, blurry pictures are the norm. Post-processing of these photos may present an avenue for salvaging blurry photos that are otherwise useless.

Objective

- Study the effects of 2 types of blur: Focus and Motion blur
- Motion: handshake motion at point of observation and motion of the observed object
- Focus: 2D Disc completly out of focus and Distance based Foreground/Background distinction
- Study the efficacy of the FTVd algorithm
- Plot the SNR and the PSNR of the images

We will use the following two images to test the FTVd algorithms.



Background and Approach

This module describes how we created our blurry image and applied the FTVd algorithm.

Background

What are the causes of the image distortion? Well using the given model for our blurry observation there are two causes: K and omega.

K is a blurring kernel. It is a matrix convolved with our original image *u* that performs a linear operation to represent the effects of a particular kind of blur. Omega is a term used to represent the additive forms of noise introduced by our camera and the environment into our imperfect observation.

To model and recover our image, we applied an algorithm know as the Fast Total Variation Deconvolution.

Fast Total Variation Deconvolution takes advantage of our problem structure and assumes several facts about the information in our image. Because of the additive noise in all of our observations, we cannot directly recover our desired image from the blurry observation by performing the inverse of operation, deconvolve our original image with the blurring kernel. Instead, we first try to minimize the noise to approximate an ideal blurring. Then we can invert the problem to find *u*.

To do this we model our problem using the following equation:

$$\min_u \sum_{i=1}^{n^2} \parallel D_i u \parallel + \frac{m}{2} \parallel K * u - f \parallel_2^2 \text{ [5]}$$

In the equation above, we have two terms: the first is our total variation norm, which is a discretized gradient measured across our entire image, the second is the data fidelity term. The data fidelity term attempts to make the difference between our blurry observation and an ideally blurred image very small. If the difference were zero, we could very easily perform the deconvolution to recover *u*. So, the minimization step will take us as close as possible to a problem with a closed form solution. This model supposes a

few facts about our problem. Primarily, it assumes that the majority of scenes in the real world will have flat, uniform surfaces. This means that our image should have very few nonzero gradients and the additive noise will introduce many random peaks and thus non-zero gradients to be minimized.

The full form of the Total Variation Regularization transforms our first model into the following:

$$\min_{w\,u} \sum_i \| w_i \|_2 + \frac{\beta}{2} \sum_i \| w_i - D_i u \|_2^2 + \frac{m}{2} \| K * u - f \|_2^2 \text{ [5]}$$

This equation adds another term to our model. Here we try to minimize the difference between the non-zero gradients and some term **w**, while simultaneously trying to make **w** as small as possible. The beta parameter in the second term helps to establish convergence, when beta is very large. For our purposes, we have used the parameters for convergence given by the FTVd reference, which has chosen optimal value for beta. We can group these terms together as the regularizing term, which constrains our model so that we have a well conditioned noisy observation.

$$\min_u J(u) = F_{\text{reg}}(u) + \frac{m}{2} \| K * u - f \|^2 \text{ [5]}$$

There are many other possible forms for constrained minimization. Different constraints will result in different ability for our model to converge. The FTVd algorithm performs this minimization using 2FFT's and one inverse FFT, giving a complexity of N log (N). In particular, we note that the FTVd will converge quickly with relatively little iteration, but it is also important to note that our problem sacrifices some clarity on textured surfaces. This algorithm is ideal for quick noise removal.

It is also important to say that this algorithm cannot function without passing in the blurring kernel. In many real world situations, such as the random motion of a handshake, it is impossible to know the blurring kernel to one hundred percent accuracy. In this case, it would be necessary to use a blind deconvolution. The process of blind deconvolution will estimate the point spread function of the blur kernel. However, in general it is always necessary to calculate the point spread function of the blur kernel. Any

signal received will be a convolution of input signal with the impulse response of the receiving system. So, to fully recover the input signal, we will need to the impulse response of our camera and any other functions that have acted on our input, namely the blur and the noise.

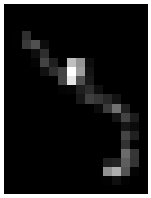
The FTVd code has been attached to this module. I has been provided open source with much thanks to Dr. Yin and his group [5].

Approach

In our project, we model four different type of real world blur, each under two possible types of noise. We break these types of blur up in three different categories and model then each with their respective blurring kernels: handshake motion, motion of the observed object, and out of focus blur.

For handshake blur, we used data collected experimentally by our Microsoft reference to develop a kernel [1]. We chose a kernel that represents a small, defined shake at the point of observation and applied it equally to the whole of the image. This data is very much similar to the linear motion function, but represents a two dimensional curve. We could have chosen to model this situation by piecing together some linear motion curves, but this more accurately represents the random motion that is possible when holding the image-capturing device. Inherently, the motion of a handshake will be unique to each observation, but in this case we have chose a specific motion blur to simplify our recovery process.

Here is our handshake kernel, represented as a black and white image:

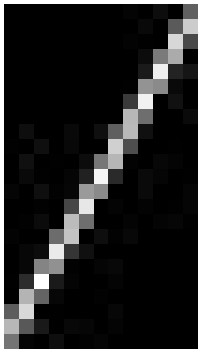


[3]

For the linear motion of an observed object, we have also chosen a determined value for the motion that will simplify our calculations. We assume that the object observed is large compared to the size of our whole

observation, such that we can blur the whole image uniformly. To develop this kernel we use the matlab function fspecial (‘motion’, amount, theta) function, which will develop a matrix of the specified distance at a specific angle [2].

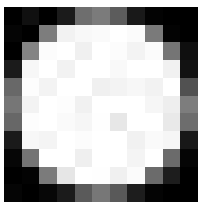
Here is the linear motion kernel as a black and white image:



To create a kernel representing out of focus blur, we used a two-dimensional disc and applied this disc to two different situations. The first represents a situation in which the picture was completely out of focus. This could correspond to the picture focused somewhere very far in the distance or infinitesimally close the camera. This results in a totally blurred image.

The next approach approximates a focal point about 10 feet behind Willy’s statue. This means that the foreground is within focus and the background out of focus. To create this image, we had to blur our image in two steps. We found that in general there is no real distance information observable in the received signal. So, we had to investigate the boundary conditions of the image to determine which areas could be considered the foreground and the background. Then our kernel could be applied only to the background of the image. We applied the FTVd algorithm to the whole image uniformly to recover the image.

This is our 2D disc representing an out of focus lens:



We applied each of these blur kernels under two different noise applications.

The first is Gaussian white noise caused by thermal Johnson-Nyquist noise or shot noise in the operation of transistors in our camera circuitry. To create this kernel we followed the model in the FTVd package, and did not used the predefined noise function. Instead, we created a matrix of random numbers drawn from a Gaussian distribution and added that to the convolution of K and u.

Next, we applied our blur kernels under L1 noise. L1 noise can be caused by random failures in sensor pixels. This can be mathematically characterized by the presence or absence of Dirac delta functions. Due to the visual nature of corruption caused, L1 noise is often referred to as ‘Salt and Pepper’ noise. We were able to add this L1 noise to our image using the matlab function imnoise.

The following image shows our handshake kernel applied with L2 additive noise PSNR=24.27 dB :

The next image shows our foreground/background distinction in the presence of L1 noise PSNR=14.94 dB:

Results and Applications
This modules provides the results and measurements of our study.

Results and Conclusions

We find that given several different real world situations, we can effectively represent their blurring effects with a point spread function. This point spread function is critical in our ability to recover our original image. In our case, we have chosen several different situations with known blurring kernels to test the ability of the FTVd algorithm.

The FTVd algorithm is effective at recovering our original image, but this is highly dependent on the amount of blur and noise present in the image. With exceeding large amount of noise corruption, we will not be able to recover our signal. However, even with large amounts of blur, we still retrieve the image as long as we can effectively create a kernel to represent the functions that have acted on our signal.

For each of our images, we measured the peak signal-to-noise ratio (PSNR), which compares the similarity of two images pixel-by-pixel. The value of one pixel in image A is subtracted from its value in image B. This difference is squared, which gives us a positive number called the squared error for the pixel. We repeat this process across all pixels in both images to find the sum of these squared errors, and divide by the number of pixels. This gives the mean squared error (MSE). If the two images are identical, the difference between each pixel will be zero for all the pixels. Dividing by the number of pixels, will give a MSE of zero.

PSNR is calculated using the following equation:
Equation:

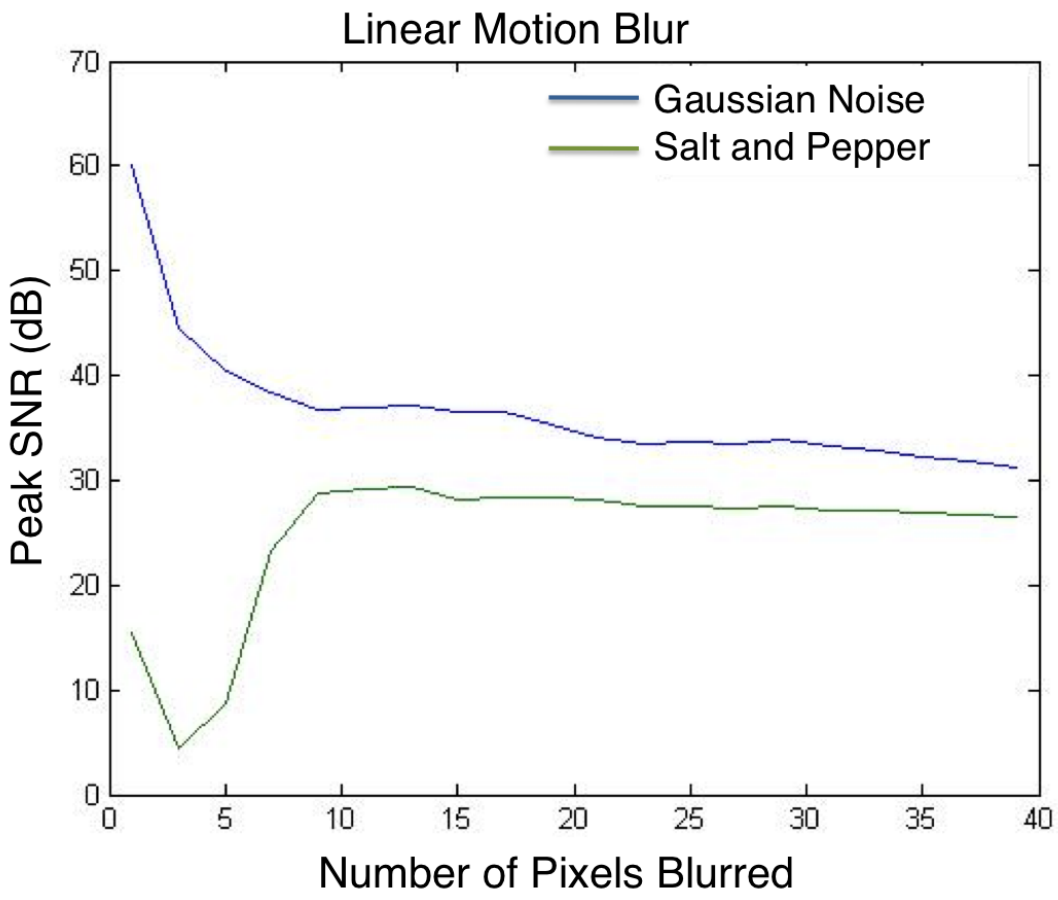
$$\mathrm{PSNR} = 10\log_{10} \frac{\mathrm{MAX}^2}{\mathrm{MSE}}$$

In our case, MAX= 1.

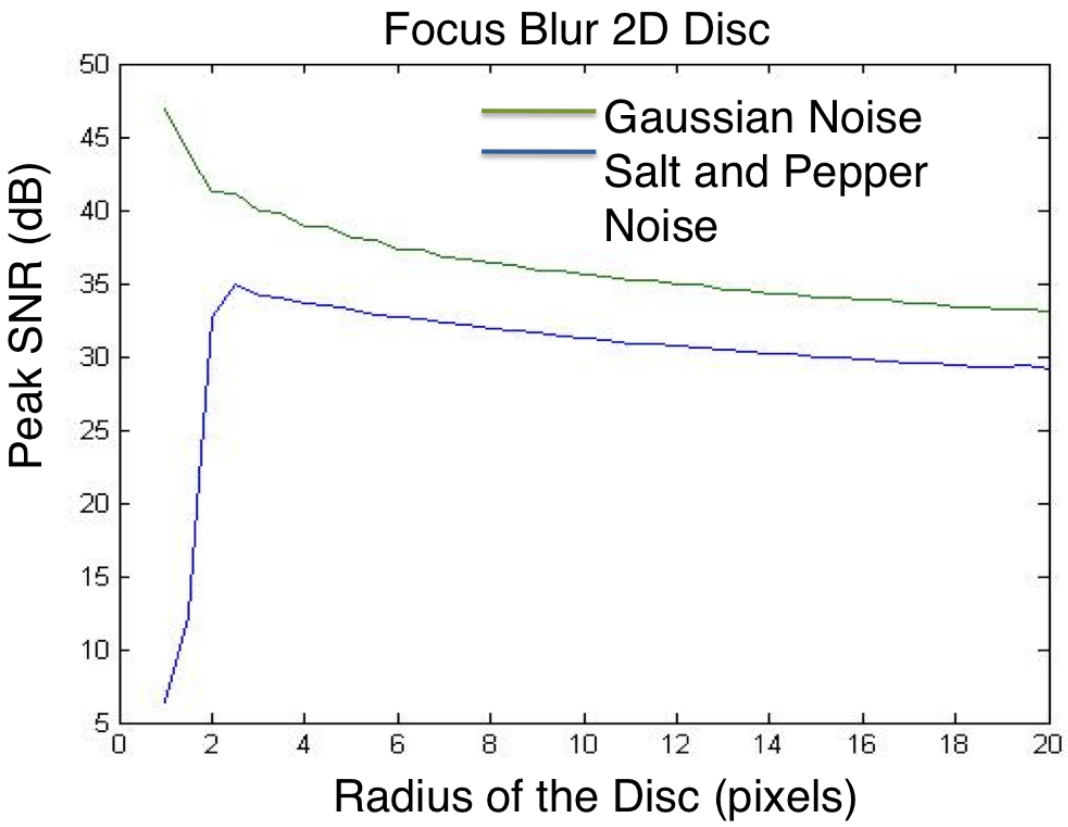
The following image shows the recovered image of handshake with L2 noise PSNR=41.97 dB:

The next image shows the recovered version of our foreground/background blur with L1 noise PSNR=34.50 dB:

The following graph shows the PSNR calculated for varying amounts of linear motion blur. It demonstrates the ability for an effective recovery every time. We can see that the PSNR of the recovered images was always with the range of about 30-50 dB, which compares favorably to the PSNR of common lossy compression ratios.



The next graph shows similar results for varied radii of our 2D disc. The graph shows resulting PSNR well within the range of 30-50dB.



The following table shows the PSNR for each all of our recovered images from their corresponding blurry and noisy observations.

	PSNR of Blurry and Noisy Observation (dB)		PSNR of Recovered Images (dB)	
Blur Kernel	L1 Salt and Pepper	L2 Gaussian Noise	L1 Salt and Pepper	L2 Gaussian Noise
Linear Motion	14.9163	23.6375	30.4995	36.5378

Handshake	14.9375	24.2680	32.1562	41.9656
Out of Focus	14.9695	28.8440	33.1344	38.0992
Foreground/ Background	15.0063	30.5060	34.5034	32.3670

Areas for Future Study

As the results show, the FTVd algorithm can be used to quickly deblur noisy images and is robust enough to handle moderate to severe amounts of blur while achieving an acceptable PSNR. Underlying these results, however, is the fundamental assumption that we have an accurate representation of the blur kernel. This is to say, the algorithm can reconstruct the clean image with some fidelity *if* we actually know or can determine the blur kernel. However, we still sacrifice some of the fine detail or sharp edges in our recovered image.

One of the methods to treat focus blur is to implement a coded aperture into the hardware of the camera lens. In this approach, a patterned mask is placed inside of the lens at the plane of the aperture. Out-of-focus photos taken with this setup will generate a specific pattern on the camera sensor, which can be used to produce a kernel.

To treat handshake blur, direct measurement of the blur kernel is possible using a high-speed camera and sensors. The problem of handshake blur was simplified in our study, as there are multiple factors that account for blur in an image. A more realistic handshake blur kernel would involve different kernels for different parts of the image. Image deblurring using these so-called “spatially-varying point spread functions” would operate much the same as in our study but would run on sections of the photo sequentially, inputting the kernel associated with the section of the photo examined at the time [3].

References
This module contains our references used in throughout our project

References:

1. Fergus, Rob, Barun Singh, Aaron Hertzmann, Sam T. Roweis, and William T. Freeman. "Removing Camera Shake from a Single Image." *Removing Camera Shake from an Image*. Massachusetts Institute of Technology, 2006. Web. 18 Dec. 2011. http://cs.nyu.edu/~fergus/research/deblur.html.
2. "Image Processing Toolbox." *MathWorks Product Documentation R2011b Documentation*. MathWorks, 2011. Web. 18 Dec. 2011. <http://www.mathworks.com/help/toolbox/images/ref/fspecial.html>.
3. Joshi, Neel, Sing Bing Kang, C. Lawrence Zitnick, and Richard Szeliski. "Spatially Varying PSFs Due to Camera Motion." *Microsoft Research - Turning Ideas into Reality*. Microsoft, 2010. Web. 18 Dec. 2011. http://research.microsoft.com/en-us/um/redmond/groups/ivm/imudeblurring/spatially_varying_psfs/index.html.
4. Veeraraghavan, Ashok, Ramesh Raskar, Amit Agrawal, Ankit Mohan, and Jack Tumblin. "Coded Aperture and Optical Heterodyning: A Mask-based Approach for Digital Refocusing and Light Field Acquisition by Conventional Cameras (SIGGRAPH 2007)." *MIT Media Lab*. Massachusetts Institute of Technology, 2007. Web. 18 Dec. 2011. http://web.media.mit.edu/~raskar/Mask/.
5. Yang, Junfeng, Yin Zhang, Wotao Yin, and Yilun Wang. "Rice University, L1-Related Optimization Project." *Department of Computational and Applied Mathematics*. Rice University, May 2010. Web. 18 Dec. 2011. http://www.caam.rice.edu/~optimization/L1/ftvd/.